Improving minBERT Embeddings Through Multi-Task Learning

Stanford CS224N Default Project

Rahul Thapa
Department of Biomedical Data Science
Stanford University
`rthapa84@stanford.edu`

Rohit Khurana
Department of Biomedical Data Science
Stanford University
`rkhurana@stanford.edu`

No Mentor; No External Collaborators; No shared project

Abstract

In this project, we extend the representational power of minBERT embeddings under a plug-and-play framework. We investigate how the performance of a pre-trained minBERT encoder model changes for three downstream prediction tasks - sentiment analysis, paraphrase detection, and semantic textual similarity, when we (1) formulate a joint, weighted loss function across the three tasks, (2) utilize regularization in our fine-tuning loss function (through the SMART objective), and (3) employ contrastive learning following the SimCSE methodology. Our best results synergize strategies (1) and (2) with separate fine-tuning on the QQP dataset followed by joint fine-tuning, achieving a mean performance of 73% on the dev and test sets.

1 Introduction

The transformer-based BERT model (Devlin et al., 2018) has revolutionized natural language processing (NLP) by providing state-of-the-art results on a wide range of tasks. Among the many applications of BERT, three canonical NLP tasks stand out: sentiment classification, paraphrase detection, and semantic textual similarity (Wang et al., 2018). These tasks form the backbone of many NLP systems, enabling sentiment analysis in customer reviews, paraphrase detection in question-answering systems, and similarity measurement for information retrieval.

Multi-task learning (MTL) has emerged as an effective paradigm for leveraging shared representations across related tasks, potentially improving generalization and data efficiency (Torbarina et al., 2023). MTL has equipped models with a better understanding regarding the general structure of human language, which is beneficial for learning harder downstream tasks. Two alternative strategies - task-specific fine-tuning and task-specific modeling - suffer drawbacks. While transfer learning via task-specific fine-tuning is common, this approach can lead to catastrophic forgetting and often fails to leverage beneficial inductive biases across tasks. Moreover, maintaining task-specific models is computationally and memory-intensive.

In this project, we design a multi-task learning BERT model based on minBERT capable of performing sentiment analysis, paraphrase detection, and semantic textual similarity (STS) assessment simultaneously. Our noteworthy modifications and contributions are as follows:

- We experiment with various settings for multi-task learning, such as combining loss functions in a weighted manner and performing back-propagation in round-robin (RR) fashion.
- We fine-tune the model first on the QQP dataset and subsequently fine-tune on other tasks, significantly boosting model performance.
- We regularize our loss function through the SMART objective (Jiang et al., 2019) and train our model through the SimCSE (Gao et al., 2021) contrastive learning (CL) framework.

- We utilize different pooling strategies, CLS and mean, and find that the latter significantly boosts model performance.

2 Related Work

Unlike previous language models trained on unidirectional or shallow contexts, BERT (Devlin et al., 2018) leverages 12 encoder transformer layers to capture bidirectional contextualized representations from large unlabeled corpora. Each layer implements multi-head attention, enabling BERT to jointly attend to information at different positions. The model outputs the sentence-level CLS token embedding as well as the individual word piece embeddings from its last layer. By training on masked language modeling and next sentence prediction tasks, BERT acquires rich linguistic knowledge that can effectively be transferred to downstream tasks through fine-tuning.

While BERT has demonstrated remarkable success, its capabilities can be further enhanced through MTL (Torbarina et al., 2023). In the context of NLP, MTL has been successfully applied to various tasks, including text classification, sequence labeling, and machine translation. By jointly optimizing multiple related objectives, MTL models can capture shared features and transfer knowledge across tasks, leading to improved performance and better generalization. However, the effectiveness of MTL may depend on the relatedness of the tasks, as tasks from different domains may exhibit learning interference and hinder performance.

To address the challenges of fine-tuning large pre-trained models such as BERT, the SMART learning framework (Jiang et al., 2019) has been proposed. Aggressive fine-tuning can often cause over-fitting, leading to the model's failure to generalize to unseen data. To combat this in a principled manner, SMART proposes (1) smoothness-inducing regularization and (2) Bregman proximal point optimization. Both sub-methods encourage the fine-tuned model to remain close to the pre-trained model while optimizing task-specific objectives. By reducing aggressive fine-tuning and promoting stability, the SMART framework has been shown to improve the robustness and efficiency of fine-tuning, leading to better performance on various NLP tasks.

Lastly, CL has been shown to improve the representational power of BERT sentence embeddings. CL encourages a model to map semantically similar sentences to nearby points in the embedding space, while pushing dissimilar sentences apart. In particular, the development of the CL framework SimCSE (Gao et al., 2021) has led to significant performance gains on tasks such as STS assessment. The unsupervised flavor of SimCSE takes an input sentence and predicts itself, using only standard dropout as a lightweight form of data augmentation. In contrast, the supervised flavor of SimCSE relies on natural language inference (NLI) datasets by using entailment pairs as positives and contradiction pairs as hard negatives. The resulting high-quality sentence embeddings generalize well to various downstream tasks.

3 Approach

As part of the CS224N default project, we first implemented a minimal version of BERT, incorporating basic frameworks such as the multi-head self-attention module and the AdamW optimizer (Kingma and Ba, 2014) with weight decay (Loshchilov and Hutter, 2017). We also implemented dropout and layer normalization, along with classification heads. We evaluated our model's performance on sentiment classification tasks fueled by SST and CFIMDB as a sanity check, achieving the metrics listed in the default project document.

Our custom modifications to minBERT follow what was delineated in 1 and 2; all improvements were coded ourselves using the associated literature references as guides. Each task, irrespective of modification implemented, was trained under a fixed architecture. For sentiment analysis, we utilized a single-layer neural network that returned five logits from sentence embedding s_1. For paraphrase detection, we utilized a single-layer neural network that returned one logit from concatenated sentence embeddings s_1, s_2, and their absolute difference $|s_1 - s_2|$ (Reimers and Gurevych, 2019). For STS assessment, we either imitated the architecture for paraphrase detection or used cosine similarity to return a post-processed, scaled logit. During our development process, we also tried using a shared layer for paraphrase detection and STS assessment given their relatedness. Empirically, this resulted in poor performance, so we opted for task-specific layers for all subsequent experiments.

During training, we utilized cross-entropy loss for sentiment analysis, binary cross-entropy loss for paraphrase detection, and mean squared error for STS assessment.

3.1 Baselines

As a baseline, we fine-tune minBERT separately on each task (given a fixed hyperparameter set) using two approaches. First, we freeze all hidden model weights and only update the linear task-specific heads. Second, we update all model parameters and fine-tune end-to-end on each specific task. These baselines represent a lower and upper bound on difficulty, respectively.

3.2 Round Robin Training

We design a batch-level MTL training routine that is generalizable across different tasks. The basic idea is to instantiate task-specific dataloaders and load a batch from each one continuously and sequentially until some termination condition is triggered. Given the imbalance in dataset size across the three tasks, with SST and STS-B being roughly equal and QQP being significantly larger, we experimented with two intuitive training paradigms.

Our first training strategy restricts each training epoch to only optimize on n samples from each task where n is the smallest dataloader size. To account for the 'lost' training data (in the case of QQP), we perform additional pre-training on a majority of QQP prior to launching equally-batched RR fine-tuning. Our rationale follows in line with transfer learning - we hope that by prioritizing learning on QQP, the model may learn useful representations that could be beneficial for the other tasks. We also played around with different sized batches (8 for QQP and 3 for SST and STS-B).

Our second training strategy allowed for complete exhaustion of the largest sized dataloader with re-instantiation of the smaller ones. Empirically, this resulted in poorer performance relative to the first strategy, so we opted for the former for all subsequent experiments.

3.3 Multi-task Loss Function

Our MTL training routine optimizes the following loss:

$$\mathcal{L} = w_1 \mathcal{L}_{\text{SST}} + w_2 L_{\text{STS}} + w_3 L_{\text{QQP}}$$

We experimented with several strategies for appropriately determining w_1, w_2, and w_3. First, we tried various fixed constants for these parameters throughout the entire training duration. Second, we adaptively modified each weight based on the model's task-specific performance in the preceding epoch. Lastly, we tried our hand at having the model simply learn what the optimal set of values are without direct supervision. Empirically, fixing $w_1 = w_2 = w_3 = 1$ resulted in the best performance, so we opted for this configuration in all subsequent experiments.

3.4 Unsupervised Contrastive Learning

To further improve the quality of minBERT's sentence embeddings, we implemented unsupervised SimCSE (Gao et al., 2021). Using a single training epoch, a batch size of 64, and a learning rate of $3e^{-5}$ (as suggested in the original paper), we iterate over batches of unlabeled sentences from the Wikipedia 1M dataset.

Each input sentence is fed into minBERT twice with dropout probability 0.3 to generate two distinct embedding representations. These pairings are considered to be our positive instances whereas all other in-batch cross-pairs are labeled as negative instances. We follow a cross-entropy objective by optimizing:

$$\ell_i = -\log \frac{e^{\text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_i^+)/\tau}}{\sum_{j=1}^{N} e^{\text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_j^+)/\tau}}$$

where τ denotes a tunable temperature hyperparameter [set to 0.05 for all experiments as suggested by Gao et al. (2021)] and $\text{sim}(\boldsymbol{h}_i, \boldsymbol{h})$ denotes the cosine similarity between dropout-induced embeddings $\boldsymbol{h}_i$ and $\boldsymbol{h}$. The preceding loss formulation encourages the model to maximize the similarity between the two representations of the same input sentence.

3.5 Supervised Contrastive Learning

To further improve the quality of minBERT's sentence embeddings, we implemented supervised SimCSE, which typically performs better than its unsupervised counterpart (Gao et al., 2021). Using a single training epoch, a batch size of 16, and a learning rate of $3e^{-5}$, we iterate over batches of labeled sentences (additionally annotated with hard negatives) from NLI datasets.

NLI datasets are comprised of premises, hypotheses, and their relationship (entailment, contradiction, neutral). Entailment pairs are considered to be our positive instances whereas all other in-batch cross-pairs are labeled as negative instances. Moreover, contradictions are incorporated as hard negatives. We follow a cross-entropy objective by optimizing:

$$\ell_i = -\log \frac{e^{\text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_i^+)/\tau}}{\sum_{j=1}^{N} e^{\text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_j^+)/\tau} + \alpha^{\mathbb{1}_i^j} e^{\text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_j^-)/\tau}}$$

where τ denotes a tunable temperature hyperparameter [set to 0.05 for all experiments as suggested by Gao et al. (2021)], $\text{sim}(\boldsymbol{h}_i, \boldsymbol{h})$ denotes the cosine similarity between paired positive or negative embeddings $\boldsymbol{h}_i$ and $\boldsymbol{h}$, and α denotes a tunable weight to assign to hard negatives [set to 1 for all experiments as suggested by Gao et al. (2021)]. The preceding loss formulation encourages the model to maximize the similarity between representations of semantically related sentences.

3.6 SMART Regularization

To mitigate overfitting during training, we implemented a regularization technique called SMART (Jiang et al., 2019). The SMART framework introduces smoothness-inducing adversarial regularization to the model by injecting small perturbations into the word embeddings while minimizing changes to the model's output. This encourages the model to output similar predictions even with noisy inputs, thereby improving its robustness to small perturbations.

Specifically, for classification tasks like sentiment analysis, the SMART loss computes the symmetrized Kullback-Leibler (KL) divergence between the original logits and the logits obtained from the perturbed embeddings. For regression tasks like STS assessment, the loss is the mean squared error between the original scalar output and the output from the noisy embeddings. To optimize the smoothness-inducing regularizer, we perform gradient updates on the noise to adversarially increase the difference between the original and perturbed logits or outputs.

To solve the smoothing objective, we employ the iterative Bregman proximal point optimization method proposed by Jiang et al. (2019). This implementation framework was adapted from the SMART torch repository. While the SMART loss function typically accepts a single tensor embedding as input, for tasks like paraphrase detection and STS assessment that involve two input sentences, we stack the corresponding sentence embeddings together. The SMART loss is then weighted by a hyperparameter (set to 0.5 for all experiments) and added to the task-specific loss during training. This regularization approach helps improve the model's generalization ability by encouraging smoother and more robust representations.

4 Experiments

4.1 Data

For fine-tuning minBERT, we utilize the SST, QQP, and STS-B datasets. SST (Socher et al., 2013) consists of single sentences from movie reviews and labels each one as either negative, somewhat negative, neutral, somewhat positive, or positive. The dataset is split into 8,544 training examples, 1,101 dev examples, and 2,210 test examples. QQP consists of question pairs and labels each one as either paraphrase or not paraphrase. The dataset is split into 141,506 training examples, 20,215 development examples, and 40,431 test examples. STS-B (Agirre et al., 2013) consists of sentence pairs and ranks each one on a scale from 0 (unrelated) to 5 (semantic equivalence). The dataset is split into 6,041 training examples, 864 development examples, and 1,726 test examples.

For unsupervised CL, we utilize the Wikipedia 1M dataset. For supervised CL, we utilize NLI datasets.

4.2 Evaluation

We utilize accuracy for sentiment analysis and paraphrase detection. We utilize the Pearson correlation coefficient for STS assessment.

4.3 Training

Both baselines (3.1) were trained for 10 epochs with a fixed learning rate of $1e^{-5}$ and dropout probability of 0.3. Across all datasets, we utilized a batch size of 8. Models were optimized through AdamW with weight decay of 0.001.

For non-baseline runs, we followed a plug-and-play approach, synergistically adopting the strategies delineated in 3. All non-baseline experiments were trained for 20 epochs with a fixed learning rate of $1e^{-5}$ and dropout probability of 0.3. Across all datasets, we utilized a batch size of 8. Models were optimized through AdamW with weight decay of 0.001. Fine-tuning in RR fashion (3.2) under these settings took around 30 minutes.

For initial pre-training on QQP (3.2), we utilized 10 epochs with 1000 iterations per epoch. For unsupervised (3.4) and supervised (3.5) CL, we utilized 1 epoch with a fixed learning rate of $3e^{-5}$ and dropout probability of 0.3. All three steps under these settings individually took around 15 to 30 minutes.

For SMART regularization (3.6), we utilized KL-divergence to calculate the SMART loss. We scaled this by 0.5 before adding it to the task-specific losses.

All models were trained on 4 NVIDIA RTX A4000 GPUs.

4.4 Results

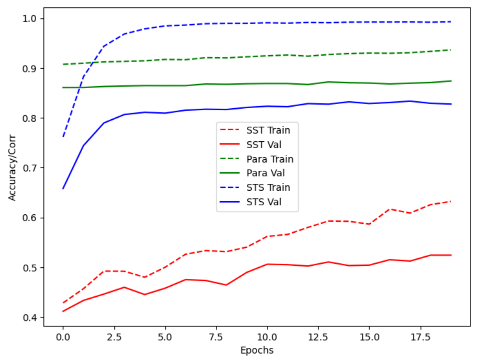
Table 1 summarizes our findings. Runs zero through three achieved an average performance of 0.74 across all three tasks and blended techniques delineated in 3. We found that mean pooling (runs 0 through 13) significantly trumps using CLS pooling (runs 14 through 23). Ablating all other modifications (run 11) besides pre-training on paraphrase detection only deteriorates performance to 0.72, indicating that 'improvements' such as joint loss optimization, CL, and SMART provide minimal gain.

The addition of initial pre-training on paraphrase detection when using CLS pooling improves performance by 0.03 (runs 14 versus 22). This inclusion naturally boosts paraphrase detection accuracy, but importantly does not deteriorate the metrics reported for the other two tasks (runs 0-21 versus 22-23), confirming our earlier hypothesis.

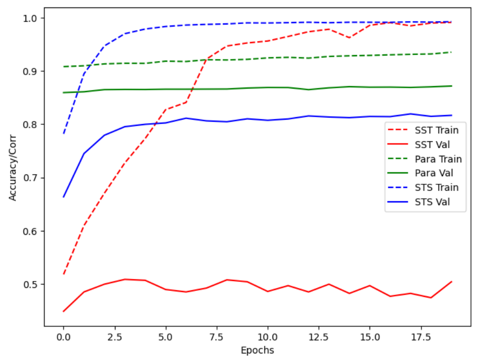
Our lower-bound baseline achieved an average metric of 0.39. Our upper-bound baseline achieved an average metric of 0.56. Our MTL minBERT model clears both baselines, with average scores of 0.74 and 0.77 on the dev and test sets, respectively. These results demonstrate the effectiveness of MTL methods in closing the performance gap between pre-training and fully fine-tuned single-task performance.

Table 1: Experimental results and ablation studies of proposed modifications

run	pre-QQP	pooling	joint-loss	SMART	U-CL	S-CL	SST	QQP	STS-B	average
0	✓	MEAN	✓	✓	×	✓	0.52	0.87	0.83	0.74
1	✓	MEAN	×	✓	✓	✓	0.52	0.87	0.83	0.74
2	✓	MEAN	×	✓	×	✓	0.52	0.87	0.83	0.74
3	✓	MEAN	✓	×	✓	✓	0.53	0.86	0.83	0.74
4	✓	MEAN	✓	✓	×	×	0.53	0.87	0.81	0.73
5	✓	MEAN	×	×	✓	✓	0.53	0.87	0.81	0.73
6	✓	MEAN	×	✓	×	×	0.53	0.87	0.80	0.73
7	✓	MEAN	✓	✓	✓	×	0.52	0.87	0.81	0.73
8	✓	MEAN	✓	×	×	✓	0.50	0.87	0.82	0.73
9	✓	MEAN	×	✓	✓	×	0.52	0.87	0.80	0.73
10	✓	MEAN	✓	×	×	×	0.51	0.87	0.79	0.73
11	✓	MEAN	×	×	×	×	0.50	0.86	0.79	0.72
12	✓	MEAN	✓	×	✓	×	0.50	0.86	0.80	0.72
13	✓	MEAN	×	×	✓	×	0.50	0.85	0.80	0.72
14	✓	CLS	×	×	×	×	0.49	0.81	0.68	0.66
15	✓	CLS	✓	×	×	×	0.51	0.81	0.65	0.66
16	✓	CLS	✓	✓	×	×	0.50	0.83	0.64	0.65
17	✓	CLS	✓	×	✓	×	0.50	0.82	0.63	0.65
18	✓	CLS	×	×	✓	×	0.49	0.83	0.63	0.65
19	✓	CLS	×	✓	✓	×	0.49	0.79	0.65	0.64
20	✓	CLS	×	✓	×	×	0.45	0.81	0.65	0.64
21	✓	CLS	✓	✓	✓	×	0.49	0.79	0.62	0.64
22	×	CLS	×	×	×	×	0.52	0.81	0.55	0.63
23	×	CLS	✓	×	×	×	0.51	0.75	0.52	0.59
Baseline (PT)	×	CLS	×	×	×	×	0.31	0.64	0.21	0.39
Baseline (FT)	×	CLS	×	×	×	×	0.51	0.79	0.39	0.56
Test	✓	MEAN	✓	✓	×	✓	0.52	0.87	0.83	0.74



(a) SMART regularization enabled.



(b) SMART regularization disabled.

Figure 1: Evaluation plots for our best model (run zero) with and without SMART regularization.

5 Analysis

The evaluation plots in Figures 1a and 1b provide further insights. Using our highest-performing configuration (joint loss, mean pooling, and supervised CL with QQP pre-training), we wanted to further examine the effects of SMART regularization. With SMART regularization, training accuracy steadily increased across epochs, reaching approximately 0.63 for SST, 0.94 for QQP, and 0.99 for STS-B (1a). Validation accuracy also increased gradually, reaching approximately 0.52 for SST, 0.87 for QQP, and 0.83 for STS-B. Without SMART regularization, significant overfitting was observed for the task of sentiment analysis (1b), with training accuracy approaching 1.0 and validation accuracy approaching 0.5. We conclude that the SMART framework is effective in mitigating model overfitting by improving generalizability and argue for its relevance in shaping the behavior of our best-performing model.

6 Conclusion

In this project, we extended the representational power of minBERT embeddings through a MTL framework by tackling sentiment analysis, paraphrase detection, and STS assessment tasks. Our experiments revealed that pre-training on QQP for 10 epochs to improve transfer capabilities before RR fine-tuning using mean pooling proved significantly more effective than using CLS pooling. Other additions such as utilizing a joint loss function, applying SMART regularization, and implementing unsupervised and supervised CL provided minor performance gains. In particular, SMART regularization reduced overfitting on the sentiment analysis task.

Though our best-performing model achieved solid results, there are certain limitations. The improvements over baseline models were modest for the sentiment analysis task, suggesting opportunities for further enhancing the MTL framework. Moreover, despite SMART regularization, our model grossly overfits to the task of STS assessment. Future work could investigate better hyperparameter tuning, more sophisticated classification heads with multiple layers, and better regularization. Lastly, expanding to a wider array of NLP tasks and datasets could provide insights into the broader applicability of these methods.

7 Contributions

R.T. contributed to writing, minBERT development, and extension implementation (SMART, joint loss, QQP pre-training). R.K. contributed to writing, minBERT development, and extension implementation (pooling, unsupervised CL, supervised CL).

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Lovre Torbarina, Tin Ferkovic, Lukasz Roguski, Velimir Mihelcic, Bruno Sarlija, and Zeljko Kraljevic. 2023. Challenges and opportunities of using transformer-based multi-task learning in nlp through ml lifecycle: A survey. *arXiv preprint arXiv:2308.08234*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.